PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06K 19/07 | A1 | (11) International Publication Number: WO 95/04328
(43) International Publication Date: 9 February 1995 (09.02.95)

(21) International Application Number: PCT/AU94/00437

(22) International Filing Date: 1 August 1994 (01.08.94)

(30) Priority Data:
PM 0294 30 July 1993 (30.07.93) AU

(71) Applicant (*for all designated States except US*): INTELLECT AUSTRALIA PTY. LTD. [AU/AU]; 1 Brodie-Hall Drive, Bentley, W.A. 6102 (AU).

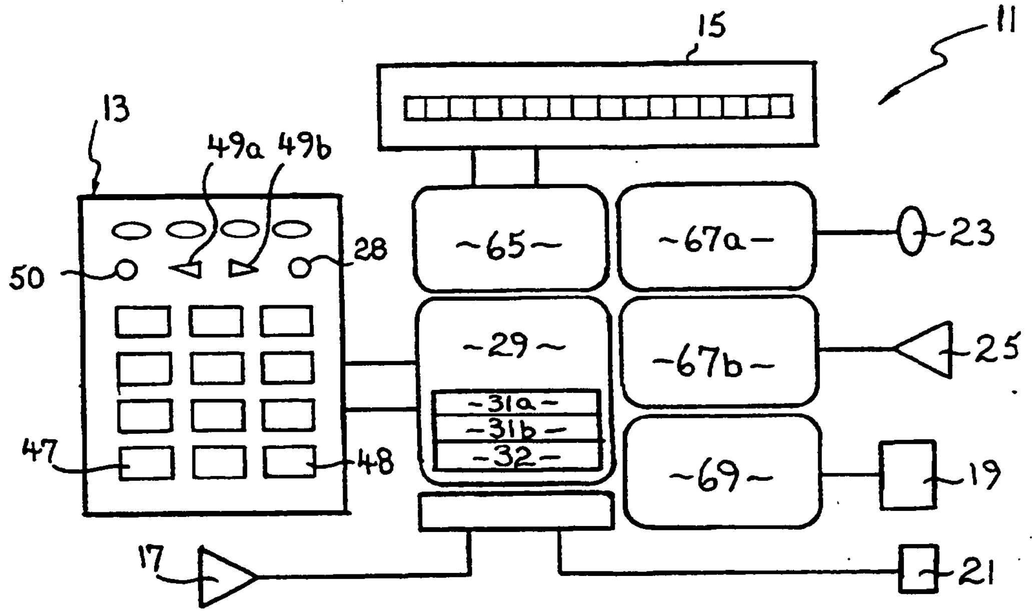
(72) Inventors; and
(75) Inventors/Applicants (*for US only*): OLIVER, Quentin, Rees [AU/AU]; Unit 1, 5 Hopetoun Street, South Perth, W.A. 6151 (AU). BERTINA, Johannes, Marinus, George [AU/AU]; Lot 111 Amherst Road, Canning Vale, W.A. 6155 (AU).

(74) Agents: HARWOOD, Errol, John et al.; Wray & Associates, 239 Adelaide Terrace, Perth, W.A. 6000 (AU).

(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, ES, FI, GB, GE, HU, JP, KE, KG, KP, KR, KZ, LK, LT, LU, LV, MD, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SI, SK, TJ, TT, UA, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD).

Published
With international search report.

(54) Title: DEVICE AND METHOD FOR IC CARDS



(57) Abstract

A device (11) for communicating with an integrated circuit (IC) card including a user interface, an external interface including an IC card interface (69) for connectedly receiving and communicating with an IC card and a microcomputer (29). The microcomputer (29) having internal storage (31a, 31b, 32) for storing data and being connected to the user interface and the external interface. An operating system (53) for operating the microcomputer (29) to control the internal storage, the user interface and the external interface. The data includes a program module (63) loaded into the internal storage for execution upon activation via the user interface. The program module (63) is imported from a said IC card or is loaded from a host to the device.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

Device and Method for IC Cards

This invention relates to integrated circuit cards (IC cards) and more particularly to a portable hand held device and a method for communicating with an IC card.

Integrated circuit cards are becoming more popular in every day use for enabling a user to perform some form of commercial transaction with a service provider. IC cards as defined by international standard ISO 7816, in particular are becoming fashionable due principally to the memory and processing power which has been able to be integrated into the design of the card with microelectronic circuit design techniques.

At present, readers and terminals for communicating with an IC card are permanently disposed and electrically interconnected at the site of the service provider or host computer connected thereto. Thus, whilst a smart card or the like can be carried on the person of a user, it is necessary for such user to actually attend the premises of the service provider or host computer in order to perform a transaction with the card by physically inserting the card into the card reader thereof and establishing the communication protocol.

It is an object of one aspect of the present invention to provide a device for enabling communication with an IC card.

It is a preferred object of said one aspect to provide a device for performing a transaction with a service provider using an IC card.

It is a preferred object of another aspect of the present invention to provide for communicating with a plurality of IC cards of different types or different manufacture, and

This invention relates to integrated circuit cards (IC cards) and more particularly to a portable hand held device and a method for communicating with an IC card.

Integrated circuit cards are becoming more popular in every day use for enabling a user to perform some form of commercial transaction with a service provider. IC cards as defined by international standard ISO 7816, in particular are becoming fashionable due principally to the memory and processing power which has been able to be integrated into the design of the card with microelectronic circuit design techniques.

At present, readers and terminals for communicating with an IC card are permanently disposed and electrically interconnected at the site of the service provider or host computer connected thereto. Thus, whilst a smart card or the like can be carried on the person of a user, it is necessary for such user to actually attend the premises of the service provider or host computer in order to perform a transaction with the card by physically inserting the card into the card reader thereof and establishing the communication protocol.

It is an object of one aspect of the present invention to provide a device for enabling communication with an IC card.

It is a preferred object of said one aspect to provide a device for performing a transaction with a service provider using an IC card.

It is a preferred object of another aspect of the present invention to provide for communicating with a plurality of IC cards of different types or different manufacture, and to handle the application specific factors of IC cards being used for different and/or multiple applications.

In accordance with one aspect of the present invention, there is provided a device for communicating with an integrated circuit card comprising-

a user interface;

an external interface including an IC card interface for connectedly receiving and communicating with an IC card;

microcomputer means having internal storage for storing data said microcomputer means being connected to said user interface and said external interface; and

an operating system for operating said microcomputer means to control said internal storage, said user interface and said external interface;

wherein said data includes a program module loaded into said internal storage for execution upon activation via said user interface, said program module being loaded from a said IC card or from a host connected to the device.

Preferably, the device includes an interpreter for executing a said program module, wherein said interpreter operates as a virtual machine on top of said microcomputer means and is adapted to prevent access to certain secured areas of said internal storage and said microcomputer means, and to prevent the direct execution of native program code of said microcomputer means.

Preferably, said interpreter comprises a compiler means for converting high level program module language into program module object code in accordance with a prescribed instruction set.

Preferably, said compiler means comprises a compiler for converting said high level program module language into program module language assembler source code, and an assembler for converting said program module language assembler source code into said program module object code, in accordance with respective prescribed instruction sets.

Preferably, said external interface includes a serial communications interface for allowing full duplex serial communications between said microcomputer means and said host connected thereto.

Preferably, said operating system provides for one or more communication options when a said IC card is connected to, said IC card interface, said IC card interface providing services with said serial interface or said user interface in conjunction with said microcomputer means to achieve said communication option.

Preferably, one said communication option comprises loading a said program module directly from said IC card for execution.

Preferably, said program module is loaded into a temporary storage area and executed upon said activation.

Preferably, another said communication option comprises passing messages between said serial interface when a host is connected thereto, and said IC card interface when an IC card connected thereto, under the control of said microcomputer means and said operating system upon activation via said user interface.

Preferably, said external interface includes a microphone for receiving tone signals remotely transmitted to the device, a speaker for generating tone signals to be transmitted from the device, and a tone signalling interface interconnecting said microphone, said speaker and said microcomputing means for decoding tone signals received by said microphone and for driving said speaker, whereby said operating system provides for a further communication option using said tone signalling interface with a telephone system upon activation via said user interface.

Preferably, said internal storage comprises a semi-permanent storage area for storing data loaded from a host connected to said serial communications interface.

Preferably, the device includes a system loader to receive and execute prescribed commands via said serial communications interface for loading said data, wherein said system loader prevents the direct execution of native program code of said microcomputer means.

Preferably, a said program module loaded from a said host connected to said serial communications interface is stored in said semi-permanent storage area, and wherein said operating system is adapted to load a said program module stored in said semi-permanent area into said temporary area for execution upon said activation.

Preferably, said IC card interface includes a routine for determining the compatibility of said IC card with the device and configuring said IC card interface for communicating with a compatible IC card connected thereto.

Preferably, said routine defines different subroutines to enable the device to differentiate between different types and makes of IC cards.

Preferably, said operating system provides for an option of a personal identification number subroutine to be performed to enable said microcomputing means to validate the identity of a user of the device before allowing a predetermined communication option to be activated by a user.

Preferably, said internal storage comprises a core area for permanently storing firmware for said operating system, said user interface and said external interface, said firmware being generally divided into three layers: the first layer comprising system and driver firmware, including said operating system and said external interface for direct hardware interfaces and firmware task management; the second layer comprising application firmware, including said user interface, for supporting the functional requirements of the device; and the third layer comprising program module interpreter firmware for supporting the operation of a said program module.

Preferably, said semi-permanent storage area also provides for storage of: system information data including a password for an initialisation loading access, an internal personal identification number (IPIN) for user access; cipher keys for cryptographic processing; program module control data; and function parameters including configuration data for the device.

Preferably, said temporary storage area is partitioned into areas for running the firmware, program module volatile data areas and intermediate storage for cryptographic functions performed by the device.

In accordance with another aspect of the present invention, there is provided a portable hand held device for communicating with an integrated circuit card comprising:-

a housing;

a keypad disposed on one portion of said housing;

a display disposed on another portion of said housing; and

microcomputing means disposed within said housing and connected to said keypad and said display for receiving input signals provided by said keypad, and sending output signals for display by said display, said microcomputing means having a computer program for operating said microcomputing means in accordance with a prescribed routine to communicate with a user of the device;

wherein a self-contained power source is disposed within said housing for supplying power to said processing means;

an integrated circuit card receptor is disposed on a further portion of said housing and connected to said processing means for receiving an integrated circuit card and communicating therewith in accordance with said prescribed routine and the input signals provided by said keypad; and

said housing is of a size sufficiently small to be carried on one's person.

In accordance with a further aspect of the present invention, there is provided a method for communicating with an integrated circuit card comprising the steps of:-

1. testing the integrated circuit card to determine whether it is one of a predetermined range of integrated circuit card types;

2. issuing a prescribed command or commands to the integrated circuit card corresponding to a particular make of integrated circuit card if said testing determines that said integrated circuit card is one of a prescribed type for communicating;

3. checking the response of the integrated circuit card to said issuing to determine whether the response is in accord with the expected response prescribed for said particular make;

4. repeating steps 2 and 3 using other prescribed commands to the integrated circuit card corresponding to other particular makes of integrated circuit if said response is not in accord with the expected response prescribed for said particular make corresponding to the prescribed command or commands until a prescribed make or integrated circuit card is determined; and

5. reading the configuration file using the prescribed commands corresponding to the particular make of integrated circuit card if said response is in accord with the expected response prescribed for said particular make to establish the parameters and language for accessing and working with the integrated circuit card.

The invention will be better understood in the light of the following description of one specific embodiment thereof. The description is made with reference to the accompanying drawings in which:-

Figure 1A is a side view of the hand held device opened in its operative position;

Figure 1B is a rear view of the hand held device of figure 1A;

Figure 1C is a front view of the hand held device of figures 1A and 1B;

Figure 2 is a block diagram showing the basic hardware architecture of the electronic circuitry of the hand held device;

Figure 3 is a block diagram showing the basic firmware architecture of the hand held device; and

Figure 4 is a block diagram of a flowchart indicating the algorithm of the communication method invoked for the purpose of differentiating between different types and makes of IC cards.

The embodiment is directed towards a multi-functional portable self contained hand held device. The device is essentially a personal information terminal which can provide data security, user authentication, telephone dialling, and act as an intelligent terminal for integrated circuit (IC) cards, in the form defined by ISO7816, better known as Smartcards. The device functions as part of a system for identification, data storage and for processing data stored internally or on a Smartcard.

The data stored internally can be in the form of special Program Modules (PM's) and/or user data, and may be for purposes such as identification, a telephone number book, etc.

Data stored on a Smartcard can be for many sorts of purposes, for example financial, medical, social security, club membership, company employee, season tickets, frequent user, etc.

The device generally includes features for:

- displaying, entering and storing of data;
- communicating to external devices via telephone, modem or direct connection;
- accessing a Smartcard;
- providing security via access control and cipher processing;
- executing a PM from internal storage or a Smartcard.

The device operates in three main configurations:

- on its own, acting as a simple Tone Dialler for telephones using Dual Tone Multiple Frequency (DTMF) or executing a PM stored internally by a service provider.
- on its own, executing a PM from a Smartcard that has been customised and personalised for a particular purpose by a service provider.
- in an extended situation, executing a PM from internal storage or a Smartcard and communicating via telephone, modem or direct connection to a service provider or host device.

As shown in figures 1 and 2 of the drawings, the device 11 is a small (shirt-pocket size) electronic device which has a housing 12, a keypad 13, liquid crystal display (LCD) 15, audio beeper 17, Smartcard reader/writer receptor 19, serial port 21, microphone 23 and speaker 25. The device is battery powered by a battery 26 disposed within a battery compartment 27, via an on/off switch 28, and operates via a microprocessor in the form of a microcomputer-on-a-chip 29 that provides secure storage and cryptographic facilities. Thus the device 11 is programmable, thereby enabling it to have many possible uses such as: a telephone list and dialler, remote user identification, an electronic purse, a low cost smart card terminal, an on course betting facility, etc.

The housing 12 comprises two planar wings 12a and 12b which are hingedly joined along their respective proximal ends. Accordingly, the wings may be opened out in an operating position as shown in figure 1a of the drawings, or alternatively may be closed together like a book in an inoperative position (not shown).

The keypad 13 is of the 'rubber mat' type and is provided with 20 keys for data entry and selection of menu choices.

The liquid crystal display 15 has one line of 16 characters and presents prompts and information to the user.

The audio beeper 17 provides a tone at 1000, 1250, 1500 or 2000 HZ and indicates keypresses and other actions.

The Smartcard reader/writer receptor 19 is designed to receive one of a variety of different types and makes of Smartcard inserted therein so that the device can read from and/or write to the Smartcard.

The serial port 21 enables the device to be connected to a modem or host device.

The microphone 23 receives DTMF tones for data reception.

The speaker 25 generates DTMF tones for dialling and data transmission.

The internal battery enables self-containment of the device and an expected service life of two years with typical use.

The microcomputer 29 in the present embodiment is a microprocessor with memory, input/output ports, timer functions, and serial communications interface, al on a single integrated circuit. The memory comprises internal non-volatile secure storage 31 for the operating program and sensitive data, divided up into read only memory (ROM) 31a and electrically erasable programmable read only memory (EEPROM) 31b, and an amount of internal volatile storage in the form of random accessible memory (RAM) 32.

Although the device can run an internally-stored program, its versatility is greatly increased by the fact that it can also run programs stored on a Smartcard. Thus with knowledge of the interface specifications and the programming language, other service providers will be able to develop their own applications and provide them on Smartcards that plug into the device 11. In addition, other organisations will be able to load their own applications into the internal storage area, thus producing fully customised devices.

There are two types of software associated with the device:

- System software, held in the ROM 31a of the microcomputer 29; and.

- Application software, in the form of Program Modules (PM's), held in the EEPROM 31b of the microcomputer 29 and on Smartcards connected thereto.

System software is common to all devices of the invention and comprises the operating system, interfaces to the various hardware components of the device and an interpreter for executing application program code. Although the system software itself is in the ROM 31a, configuration options are stored in the EEPROM 31b, so that they can be changed at loading time.

Basic user functions are incorporated in the system software and can be selected from a main menu presented at power up. The main menu offers the following options:

- Tone dialler - allowing the user to generate DTMF tones by pressing keys. This would be useful for sending tones to automatic services when only a rotary (pulse) dialling handset is available.
- Card terminal - allowing operation as a smartcard terminal. In this case the device 11 connects to another external device via its serial port 21.
- Internal PM - allowing execution of an internal PM (from a choice of up to three in the present embodiment).
- Card PM - allowing execution of an external PM on a Smartcard). Choices available will depend on what is stored on the Smartcard.

- Change PIN - allowing changing of a personal identification number (PIN) for the device. Any of the above options can be configured for internal PIN (IPIN) protection. In that case the user will be prompted for a IPIN before the option is activated.

Changing the IPIN always requires entry of the existing IPIN first.

A PM (Program Module) is an application for the hand held device 11 written in a special Program Module language (PML). There is limited storage for PMs inside the device itself, for example the present embodiment allows up to three PMs to be stored internally. Other PMs, however, can be stored on Smartcards.

The PML is a programming language that provides the usual arithmetic, logical, flow control and data transfer facilities, plus access to in-built device functions such as:

- Keypad input
- Message display
- Editing of displayed text
- Cryptographic processing
- Serial port input and output
- DTMF generating and reception
- Menu handling
- List handling
- Read/write Smartcard

When a user chooses to run a PM, its code is loaded into a buffer in the RAM 32 and executed by the interpreter.

As the device is intended to be used for financial and confidential transactions, security is important. To perform such transactions, the device 11 is required to store cipher keys and other sensitive data in a secure and non-volatile manner.

Access controls of varying levels protect data stored internally and can optionally protect access to some modes of communication and execution of PMs.

Cipher processes are provided for data security and to support cryptographic operations.

Security is provided at different levels by different system components:

- Hardware
- Operating system
- Loading system
- Interpreter

The hardware provides security by the use of the single-chip microcomputer 29 with built in features that monitor its mode of operation. If the mode is changed the internal storage areas 31 of the microcomputer will be erased.

The operating system provides security by controlling which memory areas a Program Module (PM) can access, controlling both the use of I/O features and the execution of items from the main menu.

The loading system provides security by the use of a password to start initial loading and by securing the session to reload cipher keys.

The interpreter provides security by removing the need to use the processor's native code to write programs. The interpreter limits PM access to only the memory areas allocated to it under the PML. On the other hand, if the PM was written in native code it could access any part of the memory without restriction, which could affect the security of the device.

In order to better understand the operation and implementation of the device, the firmware component of the software will now be described in more detail. The firmware is responsible for providing the user functionality of the device and control of associated electronic hardware. The firmware is masked in the microcomputer 29 of the device 11.

The firmware can be conceptually divided into three layers, namely:

(1) System and Driver firmware. This is the core firmware and is responsible for direct hardware interfaces and firmware task management.

(2) Application firmware. This layer is mainly concerned with supporting the functional requirements of the device.

(3) PM Interpreter firmware. This layer provides the programmability of the device. It draws on the above two layers to provide all device services through a PM programmer instruction set.

With reference to figure 3, generally, the system and driver firmware comprises the main menu 51, operating system 53, system services 55, and driver firmware 57. The relative architecture of the system and driver firmware, with respect to the application firmware 59, the interpreter firmware 61, and program modules 63, is shown in figure 3.

(1) SYSTEM AND DRIVER FIRMWARE

In the system and driver firmware, drivers provide hardware control sequences for the microcomputer 29, such as setting up and servicing interrupts, I/O device control and CPU timer management.

These can be classified into the following functional modules:-

(a) Liquid Crystal Display control driver 33
(b) Telephone DTMF interface driver 35
(c) Asynchronous Serial Communications driver 37
(d) Keypad Entry driver 39
(e) Smartcard interface driver 41
(f) Power on/off control 43
(g) Beeper interface 45
(h) Main menu 51
(i) Operating System 53

(a) Display Control driver

The LCD display 15 allows alphanumeric data to be displayed to the user.

Primitive operations provided for display control are:

- Initialise LCD to 8 bit, 2 line by 8 characters and 5x7 fonts.
- Read character from LCD RAM.
- Write character to LCD.
- Check if LCD is busy.
- Clear display - This clears display and returns cursor to first position on LCD.
- Display a NULL terminated string.
- Display a NULL terminated string at a logical LCD cursor address. Logical addresses have a range from zero to fifteen. The logical address allows the application to address the cursor without regard to

the underlying device address which is dependent on the LCD control hardware design.
- Move cursor to specified logical LCD address.

The hardware uses a standard intelligent LCD controller 65 for dot matrix alphanumeric display.

<u>(b) DTMF Communication Interface driver</u>

The function of the DTMF module 35 is to allow remote service support through a public telephone network without the need for a modem. The driver software 35 will support DTMF decoding using a decoder 67a and encoding using an encoder 67b. Only half-duplex communication is available, so transmission and reception can not be enabled simultaneously.

Two procedures are supplied to the application firmware 59 for this module:

- GetDtmf - Requires the maximum number of digits and the maximum waiting timeout to be specified. A buffer is also required. Returns TRUE if all the digits have been received, FALSE if a timeout has occurred.

- SendDtmf - Requires the number of digits to be specified and the buffer containing those digits.

Tone frequencies required by major public switched telephone networks (PSTNs) worldwide are supported.

Tone duration and interdigit pause parameters are stored in the EEPROM 31b. This allows customisation for PSTNs that have highly varying requirements for these parameters. The tone duration can be set to a maximum of 70 milliseconds and a minimum of 50 milliseconds. Interdigit pause can be set from 50 milliseconds to approximately 10 seconds.

Encoding requires two sinusoidal signals to be generated at the required dual frequencies for each tone representing a digit. To achieve this, two square wave signals are generated by the encoder 67b under firmware control and then electrically filtered into the corresponding sinusoids to generate a DTMF tone. The decoder 67a receives a DTMF tone and determines its corresponding digit. The result is presented as a binary number to the microcomputer 29.

<u>**(c) Asynchronous Serial Communications Driver**</u>

Asynchronous serial data communication provides a general purpose gateway to a myriad of external devices such as modems and personal computers.

This module 37 supports two data transfer modes, namely:-

- **Single byte transfer.**
- **Packet transfer.**

In single byte mode, data bytes are transmitted and received without handshaking or any framing characters.

A packet message is arranged as shown below.

STX	VL1	Data .... Data	CRC

where,

STX (02 hex) for start of transmission.
VL1 (Variable Length Indicator) number of data bytes
ETX (03 hex) for end of transmission.
CRC (Cyclic Redundancy Check) as per CCITT on the VL1, all data bytes and ETX in the packet.

Each data field consists of an 8 bit value.

When an STX is detected by the receiver, an inter-character timeout is set to a predefined value stored in the EEPROM 31b. Should the timeout occur, the receiver aborts.

Implementation of an acknowledgement protocol and transmit retry strategy is the responsibility of a PM.

A maximum of thirty two data bytes can be transmitted or received in one packet. If any more is sent, the data received up to that point is discarded and reception starts again.

The application interface 59 to this module 37 consists of:-

- InitSerial - Initialises the Serial Communication Interface of the microcomputer to 1 start bit, 8 data bit, 1 stop bit and no parity. The baud rate is required and can be set to 1200, 2400, 4800, 9600 and 19200 baud.

- GetSerial - Returns TRUE when received data is available. In single byte mode, the received character is also returned. Data is received under interrupt and is buffered in a circular queue. In packet mode, the address of the received message buffer is returned. This minimises the amount of storage allocated for messages. The synchronisation of message frames, LRC computation and checking and acknowledgements are performed under interrupts. When GetSerial returns TRUE, the message is available with framing characters and CRC stripped off. If no data is available, GetSerial returns FALSE.

- SendSerial - Requires the address of the message to be transmitted and the length of this message in number of bytes. In packet mode, it is responsible for framing and CRC computation. Timeouts are the responsibility of the PM.

- SetSerialMode - This allows the protocol mode to be changed. Changing a mode during a transmission or reception will result in unpredictable behaviour in this module.

The serial port on the microcomputer 29 is used for serial communication. It is initialised to a default baud rate, 8 data bits, 1 stop bit and no parity on power up. The default baud rate is stored in the EEPROM 31b.

(d) Keypad Entry Driver

The keypad 13 allows the user to interact with the device.

The keys are arranged as a 4 by 5 matrix. Standard key scanning techniques with debouncing are used to reliably decode keypresses.

The keypad driver 39 is responsible for scanning the matrix keypad to ascertain the row and column position of a depressed key. A scan code is calculated from this position and returned. The scan code is the linear position of each key scanning from the first key in the top left key (scan code of zero) to the bottom right key (scan code nineteen).

Debouncing is achieved by a debounce delay between scans and requires a key pressed to be released before a new key can be accepted.

The application firmware 59 accesses the key buffer with two procedures, namely,

- GetKeypad - Initiates the key scanning algorithm and returns the scan code and TRUE flag if a key press is detected. Otherwise it will return immediately with a FALSE status.

- SetKeyBeep - Allows the beeper to be enabled or disabled when a key is pressed.

(e) Smartcard Interface Driver

This module 41 implements the hardware control sequence for card power on/off, data communications and session termination as specified by ISO 7816 part 3.

Typical Smartcards that can be supported in the present embodiment are:-

- Gemplus GPM 896-Y Bit Synchronous Card (trademark).
- Gemplus MCOS16K Asynchronous Card (trademark).
- Schlumberger ME2000 Asynchronous Card (trademark).
- Schlumberger EEK2, EEK4, EEK16 Byte Synchronous Cards (trademark).

Application level firmware 59 has access to the following services:-

- System clock enable/disable - output to cards that may require a system clock input. This is also used in the reset sequence.
- Card power on/off.
- Writing data to non-asynchronous cards.
- Reading data from non-asynchronous cards.
- Card insertion/withdraw detection.

Communication to or from a Smartcard is implemented using digital port lines 69 on the microcomputer 29. All the required signals are controlled by the microcomputer and are sequenced correctly by the firmware as per ISO-7816-3 when the card is activated or deactivated. Smartcard signals are automatically deactivated when the card is removed.

__(f) Power On/Off Control__

The power up sequence requires firmware control to maintain power immediately after the power is momentarily applied by the closing of the ON/OFF switch 28. Similarly, the power down sequence is performed under firmware control, once initiated by the closure of the ON/OFF switch 28.

To effect a graceful power down sequence, this module 43 does not switch off the power immediately when the ON/OFF button 28 is pressed. Instead a power down message is provided to the Operating System 53 or Interpreter 61. A power down timeout will be set to a preprogrammed value (100ms minimum, 25 seconds maximum). This means that if the Operating System 53 or the Interpreter 61 (depending on which one is in control) does not acknowledge within the timeout period, this module 43 will simply switch off the power. Otherwise the Operating System 53 or Interpreter 61 will request this module 43 to switch off power.

An inactivity timer is also required to ensure that the device 11 cannot be accidentally left switched on for long periods of time. This value is programmable in the range of 1 to 250 seconds.

Firmware interface for this module 43 consists of:-

- Latching On power supply control. When the device is unpowered, an ON/OFF switch closure will cause this to set power control line low.
- Switching Off power supply. When in powered state, the ON/OFF will initiate the power down sequence. On a timeout or command from the Operating System or Interpreter, the power control line is set high to switch off power.
- Power down timer.
- Inactivity timer. Is cleared when a key is pressed or a DTMF tone is transmitted or received.

Additionally, PM commands can be used to:-

- Top the inactivity timer.
- Restart inactivity timer.

These can be used to prevent critical transactions with a Smartcard from being cut off prematurely.

(g) Beeper

The beeper firmware 45 is primarily for keystroke echo but is available to application level for other purposes. The beeper 17 can be activated for a specified duration in multiples of 10 milliseconds. The frequency can be selected from one of 1000, 1250, 1500 or 2000 hertz.

The application interface 59 to the beeper 17 is:

- Beep for a duration at one of four frequencies.

The beeper is controlled by an output on the microcomputer 28.

(h) Main Menu

The main menu module 51 consists of a list of alphanumeric items presented to the user on power up of the device to allow selection of system utilities.

These are:-

- The Tone Dialler.
- Smartcard Terminal for Asynchronous cards.
- Execute an internal PM. The PM to be executed is then selected with the F1, F2 or F3 keys, corresponding to PM1, PM2 or PM3.
- Execute an external PM from a Smartcard.
- Change IPIN.

Each alpha numeric item is a string, terminated by the NULL (hexadecimal zero) character. Items are displayed one at a time on the single line LCD 15. The displayed item is considered the current one. Alternatively, an invisible cursor is considered to be pointing to this item. The menu items must be stored consecutively in the ROM 31a, RAM 32 or EEPROM 31b.

(i) Operating System

The services provided by the operating system (OS) 53 are:-

- Initialisation of the hardware and volatile global data area via calls to services from all relevant modules.
- Execution of the main menu. This is the default task for the OS to run.
- Execution of the Instruction Interpreter. This is called by the OS when it is selected on the main menu.
- Maintenance of system timers.
- Provision of timed interrupt driven system management tasks such as checking Smartcard insertion and power control. This is done to avoid continuous polling by the application level firmware.

- Flow control of multiple tasks.
- Loading of parameters into EEPROM. Initiated by a START_LOAD message received on the serial communication port when the main menu is active.
- Provision of a power saving utility. This allows the application level firmware to put the microcomputer to sleep (low power mode) by making a system call.

(2) APPLICATION LEVEL FIRMWARE

In the application level firmware 59, this level of the firmware implements services that can be considered as processor independent.

It provides the following services:

- High level management of Smartcard features.
- User Interface.
- Functions to support security features of the PML.

(a) High Level Management

High level management of Smartcard features is provided by a specific firmware module referred to as the Smartcard Manager 71. In the present embodiment, the Smartcard Manager 71 provides for:

- ISO pass through mode for Asynchronous Smartcards which conform to ISO 7816-3.
- PM instructions to perform emulated ISO Smartcard commands. These are GetISO and SendISO. GetISO issues the ISO command header and waits for data to be read from the Smartcard. Conversely, SendISO issues the ISO command header with additional data if required.

- System routines to select a directory, read a director, select a file and read from a file for MCOS116K Smartcard.
- System routines to open a file and read a file for a ME2000 Smartcard.
- System routines and corresponding PM instructions to present secret code, write and read and erase memory for GPM 896-Y Smartcards.
- System routines and corresponding PM instructions to read from and write to EEK2, EEK4, EEK16 Smartcards.

The Smartcard Manager module 71 uses the services provided by the Smartcard Interface driver 41 and the Asynchronous Communication driver 37.

In order to determine the type of Smartcard inserted into the Smartcard reader/writer receptor 19, the Smartcard Manager 71 invokes a routine which selects between one of the card specific subroutines for communicating with the particular Smartcard using the appropriate command codes, data format and transfer protocols specific to the particular Smartcard. As these vary between different types and makes of Smartcards, the initial routine needs to identify the particular Smartcard before it can select the appropriate subroutine. This routine will be described in more detail later.

The following sections summarise the characteristics of some smart cards which can be supported in the module of the present embodiment.

(i) ISO Smartcard Message Header
For asynchronous Smartcards, an ISO pass through mode is supported. In this mode, the device becomes a transparent serial link between an external terminal and a Smartcard.

The protocol use is type T=O asynchronous half duplex transmission as defined in Section 8 of ISO 7816-3 [2]. This is supported by the ME2000 and MCOS16K Smartcards. The command message header is shown below.

CLA	INS	P1	P2	LEN

Where CLA = 0, INS = instruction, P1, P2 additional data to complete instruction, and LEN = number of data bytes to follow this command.

(ii) MCOS16K Gemplus

The main characteristics of this Smartcard are:-

- Uses the ISO 7816-3 protocol.
- Data or PM has to be stored in files.
- Files of the same type are clustered under a directory.
- A complex system of secret code and access control management provides high level of security to stored data and PM.

Services provided by the Smartcard Manager 71 for this card are:-

- ISO Pass Through.
- Selecting a directory (with directory number, 0-30).
- Selecting a file (with a file number, 0-255).
- Reading data from a file.

The last three services are meant to be used by the Operating System 53 to load a PM from this Smartcard. Writing data and all other Smartcard functions can be accessed with the ISO Pass Through service. PM programmers are encouraged to use the ISO Pass Through service exclusively.

(iii) ME2000 - Schlumberger

Main characteristics:-

- Uses ISO 7816-3 protocol.
- Data and PM are stored in files.

Services provided are:-

- ISO Pass Through.
- Opening a file (Specified by number 1-255).
- Reading data from an opened file.

The last two services are meant to be used by the Operating System 53 to load a PM from this Smartcard. Writing data and all other Smartcard functions can be accessed with the ISO Pass Through service. PM programmers are encouraged to use the ISO Pass Through service exclusively.

(iv) Schlumberger EE2K, EE4K EE16K

Main characteristics:-

- Dumb card consisting mainly of EEPROM with I^2C synchronous protocol.
- Free format data.
- Card sizes of 256, 512 and 2048 bytes are supported.

There is no protection code implemented by this card.

Services:-

- A multi-byte read sequence starting at a specified byte address.

- A multi-byte write sequence starting at a specified byte address.

These are made available as PM instructions.

(b) User Interface

The User Interface is provided for by a specific firmware module 73 which uses the LCD and keypad drivers 32 and 39 respectively, to provide an interface consisting of:-

i) Scan to Output Keycode conversion.
ii) Entry field editing functions such as alphanumeric entry, cursor control and user prompting control.
iii) A menu system.
iv) A list handling routine.

(i) Scan to Output Code

The keypad driver 39 returns a key entry as a scan code which is a representation of the row and column for that key. This needs to be converted to an output code for display. Conversions are performed through a look up table of scan codes against output codes. In CCITT mode (see Data Entry, Mode and Editing Control), output codes are also dependent on the number of times a scan code has been successively read by the driver. For example, pressing the same key twice will cause the second layer of the output code table to be used for scan code to key output code conversion. To select one of these multi-layered output codes, the period key 47 is pressed. This is known as the "current alphanumeric accept" key in CCITT mode.

The first layer of the key output code table is located in the EEPROM 31b so that customised key output codes can be loaded. The next three layers required for a CCITT keypad are stored in the ROM 310.

These CCITT codes consist of the numerals zero to nine, the alphabetics and two control characters ('#' and '*'). The device does not support the two control characters.

PMs can be written to read these scan codes and/or output codes and take any arbitrary action based thereon.

(ii) Data Entry, Mode and Editing Control

The user can enter data in one of three modes, namely:

- normal mode - the keypad 13 normally.
- extended alphanumeric mode - this supports the complete alphabet by assigning more than one key output code to each key. Pressing the same key more than once in succession will scroll through the superimposed codes for that key. Pressing a new key will enable the last key code to be accepted. This is known as the CCITT mode.

- PIN entry mode - this causes each (numeric) keypress to be echoed on the screen as an asterisk. Only numeric keys and the E(nter) key 48 are accepted, as indicated by a beep from the beeper 17. Othyer keys are ignored.

For prompted entry of alphanumeric data, a GetInput routine is specified with the following behaviour:-

- The prompt string appears on the left hand side of the display 15 and the input field is concatenated to the prompt string. A blinking cursor is activated on the leftmost character position of the input field. A blank space is automatically inserted between the prompt and the input field.
- A maximum field width (in characters) needs to be specified.

- Numeric only, alphanumeric or PIN entry mode can be specified for the entry field. When the cursor is at any entry field position except the last one, entering a character will cause the character to be displayed at that position and the cursor moved to the next position to the right. In the last position, the character is displayed without moving the cursor. Any further valid character entered will replace it.
- The left arrow key 49a functions as a backspace key. This moves the cursor one LCD location and deletes that character. However, if the cursor is at the last position in the entry field, it deletes the character at that position and remains there. For backspace to work, the cursor must be placed at the end of the input string entered up to that point.
- If the enter key 48 is pressed, it will return with a USER_DATA flag. The data entered by the user is contained in the buffer supplied to it. This is a string of zero or more ASCII characters terminated by the NULL (Oh) character.
- Otherwise, pressing the cancel key 50 will cause it to return a USER_CANCEL flag.

The data entry function can be modified by a NO_DISPLAY flag to display an asterisk for each character entered. No cursor control is available and the entry mode is strictly numeric only.

(iii) Menu System

The menu routine is invoked to enable a user to operate the main menu 51. The menu responds to the following keystrokes:-

- The left arrow key 49a - this causes the menu item previous to the current one to be displayed. If the first (zero) item is displayed when the key is pressed, the selection will roll over to the last item.

- The right arrow key 49b - this has the reverse effect of the left arrow key 49a. It causes a rollover to the first item when the current item is the last one.
- The cancel key 50 - this causes the menu routine to terminate and return with a NO_KEY flag.
- The enter key 48 - this causes the menu routine to return with the index of the current menu item.

The menu handling utilities are also accesible by a PM. A PM can load items into the device for menu options, or use items held in permanent storage (i.e. in ROM). Thus, in conjunction with logical PM instructions, the menu routine allows a high level method of handling menu interaction with the user.

(iv) List Handler
A list handler is specified to provide a high level method of browsing through a list of items and the fields in a list item. In addition, a field can be selected for editing. In this mode, the editing functions are identical to that for data entry. The list handling routines are available to a PM. Each item can have up to four alphanumeric fields.

This list handler behaves in the following manner:-

- When first invoked, it allows list items stored in the RAM buffer 32 to be scanned in a forward or backward sequence with the left (UP) and right (DOWN) keys 49a and 49b. Unlike the menu routine, there is no rollover feature for list scanning. If an UP arrow key 49a is pressed when the cursor is pointing to the first item, the handler will return with a UP_ITEM code. Similarly, a DOWN_ITEM code is returned when a

DOWN key 49b is pressed with the cursor pointing to the last item. These return codes can be used by a PM to implement its own rollover on the current portion of the list or to get more items of the list from a Smartcard or internal storage when the user tries to go beyond the buffer boundaries.

- If the Enter key 48 is pressed at this point, the handler returns the current item number. Additionally, a FIELD_CHANGED code is returned if at least one field has been updated. Otherwise, a FIELD_OLD code is returned.

- If the Cancel key 50 is pressed, it returns a NO_ITEM code.

- If the Edit key 47 (which takes over the function of the period key in list mode) is pressed, the current item is selected and its fields can be scanned. The right and left arrow keys will cause the next or previous field to be displayed.

- Pressing the Enter key 48 on a field will cause the field to become editable. A blinking cursor at the last character position of the field will be displayed.

- Pressing the Enter key 48 again will accept the changes made.

- Pressing the Cancel key 50 will exit field editing mode and allow the fields of the item to be scanned again.

- Pressing the Cancel key 50again will allow list items to be scanned.

- Pressing the Enter key 48 or Cancel key 50 will exit the list handler as previously described.

It is the responsibility of the PM to perform further processing of the relevant UP_ITEM, DOWN_ITEM, FIELD-CHANGE, FIELD_OLD or NO_ITEM return code. When the list handler is exited, an indicator signals the PM if any item has been changed.

(c) Security Features

Security management is achieved for the functions performed by the device firmware by providing:-

- well defined access control of storage areas such as configuration data in the EEPROM 31b, PMs in the execution buffer of the RAM 32, and RAM storage of intermediate results from the running of cryptographic utilities.

- Cryptographic utilities, an IPIN and a system password.

- Secure loading sequences through the asynchronous communication port 21 for programs, data, parameters and cipher keys stored on the EEPROM 31b. In the present embodiment, except for the IPIN, all other EEPROM parameters can only be loaded through the serial port 21.

These features will now be described in more detail.

(i) EEPROM Image and Access Control

The layout of the EEPROM is summarised in Table 1 below.

B7FFh	Checksum (2 bytes)	EEPROM Map Checksum Area
	Device Id (4), Issuer Id (4) Key Output Code Table (19)	Permanent Information Area
	Baud Rate (1) Dtmf tone duration (1) Dtmf interdigit pause (1) Inactivity timeout (1) Power Down timeout(1) Card Interdigit timeout(1/2) Serial Inter-Character timeout(1/2)	Function Parameter Area
	User Data(140)	
	PM data areas	
	PM code areas	
	4 set of Cipher keys(32)	
B600h	Password (8) IPIN (4) IPIN Toggle and Options(1) Allocation Table for three PMs. (12) Retry Counter(1)	System Information Area

Table 1 EEPROM Configuration

The EEPROM is divided into the following areas.

System Information area. This contains:-

- A one byte Retry Counter for IPIN locking/unlocking. The maximum count is four consecutive unsuccessful IPIN presentations.

- A twelve byte allocation table, as shown in Table 2. Each four byte entry contains control information for a PM stored in the EEPROM 31b. The low nine bits of the first word encodes the size of the PM data area in the EEPROM. Bit 9 when cleared (0) gives this PM access to the User Data area. The low nine bits of the second word encodes the size of the PM code area. A correct IPIN presentation is required to run this PM if bit 9 is cleared.

B609h	PM 3 code size PM 3 data size
B605h	PM 2 code size PM 2 data size
B601h	PM 1 code size PM 1 data size

Table 2 PM allocation table

- IPIN. A two byte storage for a four digit PIN. The first digit entered is stored at the lowest address (B60Dh).

- IPIN toggle and Option byte. Starts at B60Fh. The User Data area is defined to exist if bit 7 of this byte is cleared (0). Similarly, bit 0 to 2 (when cleared) specifies IPIN protection for "Tone Dialler", "Smartcard Terminal" and "Execute PM from Smartcard" items of the Main System Menu.

- An eight byte password. The starting address for this is B610h.

Cipher Key area. This contains four sets of cipher keys, namely, key 0 to key 3. Each key is eight bytes long. Cipher key 0 is located at address B618h.

PM Code areas. Up to three code areas can be defined (in allocation table as shown in Table 3).

PM Data areas, as also shown in Table 3. Up to three data areas can be defined.

	PM 1 Data
	PM 2 Data
	PM 3 Data
	PM 3 Code
	PM 2 Code
B638h	PM 1 Code

Table 3 PM code and data areas

User Data area. This consists of 140 bytes of user defined data.

Function Parameters area. Contains modifiable parameters that determine the behaviour of PM instructions.

These are:-

- Inter character timeout indicator for serial communication reception. This contains a multiplication factor for the shortest character time (at 19.2 kbaud). This minimum delay is arbitrarily set to 5/19.2 milliseconds (or 5 character delay).

- Inter byte timeout in multiple of 10 milliseconds for receiption from an Asynchronous Smartcard. This is required when receiving the answer to reset and in the ISO pass through mode.

- Power Down timer. Set in multiples of one second.

- InActivity timeout. Set in multiples of 100 milliseconds.

- DTMF interdigit pause between transmission tones. Set in multiples of 50 milliseconds.

- DTMF tone duration. Additional tone duration to basic period of 50 milliseconds. The resolution is 5 milliseconds. The maximum duration is limited to 70 milliseconds.

- Baud Rate. This is the bit pattern for the BAUD register of the microcomputer. Baud rates supported are 1200, 2400, 4800, 9600 and 19200.

Permanent Information area. Consists of:-

- Key Output Code Table. Nineteen codes are stored here. The relative position (from the first code stored at lowest address) is the scan key code produced by the keypad scanning firmware. Given a scan code, the keycode can be read off this table.

- Issuer ID. Four byte code that identifies the device EEPROM map issuer.

- Device ID. Four byte serial number for each unit. An eight digit number can be encoded.

EEPROM Map Checksum area. Contains the 16 bit addition of the previous 510 bytes of EEPROM data. Read and verified by the OS before activating the interpreter and on power up. If an error is detected, no PM can be run. Only the loader can be activated.

(ii) RAM Access Control

The microcomputer RAM 32 is partitioned into areas for the firmware, PM volatile data areas and intermediate storage for cryptographic functions. These memory address partitions allow the interpreter to easily check invalid access to the cryptographic processing area during runtime.

(iii) DES Encryption/Decryption Utilities

This implementation conforms to AS2805 Part 5.

Supported DES functions are:

- Encryption of an 8 byte block with an 8 byte key.

- Decryption of an 8 byte block with an 8 byte key.

- Chain encryption of an 8 byte block with an 8 byte key.

- Chain decryption of an 8 byte block with an 8 byte key.

(iv) Configuration and Loading

In order for the hand held device 11 to be useable, it is necessary to load configuration settings, security information and PMs to the device from a host device or system. PMs can also be loaded to a Smartcard. All of this is performed by direct commands sent to the device via the asynchronous serial communications port 21.

In the present embodiment, the following configuration parameters can be set by specific commands:

- IPIN protection for individual main menu items.
- IPIN protection for individual internal PMs.
- Inter-character timeout delay for serial communications.
- Inter-byte timeout delay for reception from an asynchronous Smartcard (needed when operating as a Smartcard terminal).
- Power down timer. This determines the maximum delay from when the power off button is pressed to when the unit actually shuts down, allowing an active process time to conclude gracefully.
- Inactivity timer. If there is no activity for this period, the device will automatically power down.
- The duration of generated DTMF tones and the interval between them.
- Default serial communications rate. The rate can be changed by a PM.

In addition, the following items of security data can be loaded with specific commands:

- Four cipher keys, each eight bytes long.
- An eight byte password, required for re-loading.

With regard to internal PM's, as previously mentioned, the device can accept up to three PM programs, each with a separate data component. There can also be a common data area available to all PMs.

In order to execute these commands and still maintain security over memory access, a System Loader is provided in the firmware to support these commands. The system Loader is responsible for processing loading commands sent by an external device (usually a personal computer), and so avoids the need for using the native instructions of the microcomputer for loading purposes. Furthermore, it prevents program code written in native instructions from being executed, which would otherwise have the ability to access the entire memory range of the microcomputer.

In order to initiate loading, an Activate Loader command must be sent to the device after power has been applied and prior to any other action being performed. The Activate Loader command will put the device into the System Loader state. In this state, the unit is controlled by loader commands - access to the normal device functions is not possible until the unit is re-powered. The initial Activate Loader command will check the integrity of the EEPROM and erase either all of the EEPROM if the lower checksum verification failed, or the main area of EEPROM (ie. the area above the lower checksum) if the main checksum verification failed. The Activate Loader command may be resent at any time to determine the current loader status.

After a successful 'Active Loader' command has been issued to the device, the following loader functions can be performed.

Password Sign-On

A sign-On with a correct password is required before the device will permit any loading operation except for Cipher Key Reload (which requires a Terminal Sign-On). In the present embodiment, if an incorrect password is presented four times, the device self secures, requiring an initialisation load to be performed. A correct password presentation resets the password retry counter. A successful Sign-on resets the IPIN retry counter, so it unlocks a locked device.

Performing a Sign-On starts a loader session. The session is terminated with a Sign-Off command.

Initialisation Load

Initialisation clears all EEPROM contents and programs the entire EEPROM space. It is required on a virgin unit or one that has self-secured. All data stored in EEPROM can be set as required, including password, cipher keys, ID values, configuration and PMs.

Update Load

An update load clears and programs all EEPROM space except the system area, which contains the password, cipher keys, Device ID, User Id and key output table. The load covers the configuration parameters and PMs and their associated data (if required). In practice, PM data would probably be left at all 0xFF for this load, with PMs being loaded.

PM Load

This loads one or more PMs and the associated data if required.

Change Password

The password can be changed during a loader session.

Terminal Sign-On

A Terminal Sign-On involves presentation of a random value sent both in the clear and encrypted under cipher key 0 (the Cipher Reload key). If this validates successfully, Key Load operations can be performed.

Key Load

Cipher keys 0 to 3 can be loaded at any time after a successful terminal Sign-On. However, the Cipher Reload key (0) can only be changed once from its erased value of 0xFFFFFFFF.

Smartcard Loading

A separate PC program is used for Smartcard loading. This loader correctly formats a Smartcard, depending on its type, then loads one or more PMs to it in a defined format that can be read by the device.

A device can be set to Smartcard terminal mode and connected directly to a PC running the loading program. The PC sends ISO commands to carry out the loading; these are passed through to the Smartcard.

(v) IPIN Usage and Activation

The IPIN controls access to services provided on the main menu of the device. Certain items on the main menu can optionally be protected by the IPIN, meaning that the item will not be useable until a valid IPIN is entered. Additionally, the "Change IPIN" item is always protected by the current IPIN, meaning that the current IPIN must always be entered first, before the IPIN can be changed.

A retry counter is associated with the IPIN. It keeps a count of the number of consecutive incorrect IPIN presentation. When the count reaches four, IPIN protected services are locked out. To recover from this, the password needs to be correctly presented to the device and a "Reset PIN Retry Count" command issued. The retry counter is always cleared on a correct IPIN presentation.

IPIN Toggle and Option byte is kept in the EEPROM, image and can only be updated through an initialisation load as previously described.

(3) INTERPETER FIRMWARE

In the interpreter firmware 61, the interpreter will implement two main categories of PM instructions, namely:

- General purpose - These are instructions that allow arithmetic and logical operations on unformatted one and two byte user defined data areas. In addition, some of these will allow flow control of the PM. These include jump and subroutine call instructions. General purpose instructions are similar in syntax and semantics to those provided by the microcomputer 29.

- Special purpose - These are related to specific features of the device. Included are instructions to access hardware functions such as DTMF dialling, serial communications. Generally, these instructions are tied to corresponding firmware functions in both application and device driver layers.

The interpreter is implemented as a virtual machine that resides on top of the microcomputer 29. It is similar to the microcomputer core in that it has its own registers and instruction set. A detail description of the instruction set provided by the interpreter will be described in more detail later.

The security of the device is enhanced by executing a PM through the interpreter. The interpreter prohibits access to secure memory areas by PMs, by observing memory access control rules described in more detail later. It also prevents program code written in native instructions to be executed by the microcomputer which would otherwise have the ability to access the entire memory range of the microcomputer.

The PML consists of an instruction set allowing a PM application to be tailored for individual cases. This is an interpreted instruction set.

As previously described, the interpreter implements a virtual machine. This virtual machine consists of:

- Instruction address counter. Also known as program counter (PC).
- A return address register stack to implement pseudo functions (functions without inbuilt parameter passing or local (stack) variables).
- A status register. Zero, positive/negative or overflowed result due to latest arithmetic and logical operation.
- A register set that is a subset of the microcomputer 29. These consists of two 8 bit registers (A and B) and two 16 bit registers (IX and IY). Registers A and B can be concatenated to form the D register.

The PML instructions will be executed sequentially until a function call or a new address is loaded, e.g. a JMP instruction is executed. Parameter passing is via memory locations defined by the user in the global area.

The instruction set consists of special function instructions and general purpose instructions. These are listed in the following sections.

(a) Special Functions

These instructions are concerned with supporting the hardware directly and high level user functionality.

InitSerial(baud, type) - Initialise serial UART with the specified baud rate. Parameter type specifies if a byte by byte or message packet data encoding is used.

GetSerial(data, num) - Returns number of bytes read and stored in data. Parameter num specifies the maximum to be read. An inter character timeout is associated with this instruction so that it does not wait forever.

SendSerial(data, num) - Put num bytes of data in low level transmit buffer to be transmitted by serial port handler.

AddPhrase(address, index) - Adds a string in the phrase book (via index) to a string at address. If this address does not contain a string, it must at least hold the NULL character (0 hex)

AddString(string1, string2) - Adds string2 to string1.

SetDelay(duration) - Set a delay for a duration in milliseconds.

ClearLCD - Clear LCD and set cursor to home position.

GotoLCD(line, column - Move cursor to line and column.

WriteLCD(character) - Write the character at the current LCD location.

WriteString(string) - Write a null terminated string to the LCD starting at the current cursor location.

GetDTMF(data, num) - As for GetSerial but for DTMF port.

SendDTMF(data,num) - As for SendSerial but for DTMF port.

GetInput(prompt, buffer, fieldwidth, echoMode) - Display a prompt with a following input field. If echoMode is set to ECHO, data entry is displayed on LCD and editing functions are made available. Otherwise, asterisks are displayed and all editing functions are disabled. The data entered are kept in buffer.

CreateMenu(menuHeader, itemList, numItems). Setup appropriate values in menuHeader, given the start address of list of items and the number of items. The menuHeader parameter is used by the RunMenu command.

RunMenu(menuHeader) - Controls menu interaction with the user. Returns with the menu item number if a valid selection has been made. Otherwise, a NO_KEY token is returned.

CreateList(listHeader, numFields, numItems, listItems) - Creates a list header, given the other parameters. This is used by the RunList command.

RunList(listHeader) - Controls list interaction with the user.

SendISO(startFlag, Ins, P1, P2, length, data) - Write to Smartcard with ISO protocol. In general the startFlag is zero. Ins defines the operation to be performed by the Smartcard. Data bytes of length of zero or more bytes can be attached.

GetISO(startFlag, Ins, P1, P2, length, data) - Read from Smartcard with ISO protocol.

Beep(frequency, duration) - Beep at frequency for duration milliseconds.

Encrypt(data, result, cipherKey) - DES encrypt eight bytes of data and store in result. cipherKey is an index to the selected key in cipher key table.

Decrypt(data, result, cipherKey) - DES decrypt instruction.

SetICV(icv) - Set the initial chaining vector for a subsequent MAC or CBC operation.

EncryptCBC(data, result, cipherKey) - Cipher Block Chaining (CBC) encryption. Eight bytes of plain text is pointed to by data and chain encrypted with cipherKey. Note that the result is returned on every completion of this routine. The initial chaining vector is set by **SetICV**.

DecryptCBC(data, result, cipherKey) - Cipher Block Chaining (CBC) decryption. Eight bytes of plaintext is pointed to by data and chain encrypted with cipherKey. Note that the result is returned on every completion of this routine. The initial chaining vector is set by **SetICV**.

CalcMAC(data, numBlocks, result, cipherKey) - This generates a Message Authentification Code (MAC) by performing a EncryptCBC and masking out the least significant 32 bits of the result.

ReadIPIN - Reads a pin of 4 digits from the keypad and compares with IPIN. Returns TRUE or FALSE.

SCBitRead(bitAddress,numBits, buffer). Read a specified number of bits starting at specified address for GPM 896-Y Smartcard (trade mark).

SCBitWrite.(bitAddress,numBits). Write a specified number of zero bits starting at specified address for GPM 896-Y Smartcard (trade mark).

SCBitErase(bitAddress). This writes a one at the specified bit address. The result is dependent on the area where the one is written to. Only for GPM 896-Y Smartcard (trade mark).

PresentCSC(csc). Present Card Secret Code for GPM 896-Y Smartcard (trade mark).

EraseAppl(esc, areaNum). Erase area specified by Erase Secret Code and area number. Only for GPM 896-Y Smartcard (trade mark).

ReadBytesEE(start, num, data). Requires start address, the number of bytes and a pointer to a data buffer.

WriteBytesEE(start,num,data). Requires start address, the number of bytes and a pointer to the data bytes.

ME2_OpenFile(name). Parameter "name" is a byte value, excluding 0 and 255. This uniquely identifies a file on the ME2000. Only "ordinary" file type is supported.

ME2_Read(numBytes, buffer). Read specified number of bytes from the ME2000 Smartcard into the buffer.

M16_ReadFile(fileNum, offsetInFile, numBytes, buffer). Reads "numBytes" of data from file "fileNum" starting at offset "offsetInFile" into a buffer. This operation is defined only for files with plain access modes. It will not work for file requiring ciphered access.

M16-SelectFile(mode, fileType). Selects a file specified by "fileType" (0 to 255) and make it current. Parameter "mode" (0 to 3) specifies the search mode to be used to locate the file in the current director.

<u>**(b) Flow Control**</u>

These instructions are concerned with controlling the flow of program execution.

label : instruction - defines a label at the address of an instructions.

Jmp absolute address - jumps to address.

Rts - return to calling routine.

Jsr label - gives control to function code starting at address indicated by label. The return address is saved on the stack.

Bne label - Branch to label if NOT EQUAL.

Beq label - Branch to label if EQUAL..

Bcc label - Branch if NOT OVERFLOW.

Bcs label - Branch if OVERFLOW.

LoadPM - Load a PM from internal EEPROM or Smartcard into the PM execution buffer in RAM. A PM constant identifier defines the PM, i.e. PM1, PM2, PM3 or PM_SMC for PM in Smartcard.

JumpPM - Restart execution of a PM starting at address zero of the execution buffer.

ExitPM - Exit from a PM back to the System Main Menu (Refer to Section 5.6).

(c) Arithmetic and Local Instructions

These allow simple arithmetic and bitwise operations to be performed on memory locations.

Add[**A** or **B** or **D**] mem - Add two 8 bit values if A or B register is specified. Otherwise a 16 bit add is performed. One of the operand is the implicitly specified register and the other being "mem" memory. Result stored in A or B or D Register. Sets the overflow flag in the status register. Seen as a carry.

Sub[**A** or **B** or **D**] mem - Subtract "mem" from the specified register. Result in register. Sets the overflow flag. Seen as a borrow.

Asl mem - Shift the bits in mem left. The least significant bit is lost and a zero bit is shifted into bit seven. Sets the zero flag if result is zero.

Asr mem - Shift right. Reverse effect of ASL. Sets the zero flag if result is zero.

And[**A** or **B**] mem - Bitwise ANDing of register with "mem". Result in register. Sets the zero flag in the status register.

Ora[**A orB**] mem - Bitwise ORing of register with mem. Sets the zero flag if result is zero.

Cpl - Bitwise complement of D register. Sets the zero flag if result is zero.

Com[A or **B]** - Bitwise complement of A or B register. Sets the zero flag if result is zero.

Cmp[A or **B**] mem - compares register with "mem". Sets the zero flag if result is zero. Sets the positive flag if mem1 is greater than mem2. Otherwise clears the positive flag.

(d) Memory management Instructions

A simple memory allocation is defined for PML application variables and non-volatile storage. The RAM area is divided into a system area used for internal data manipulation and a user area which is directly available to PML applications. Similarly, a user EEPROM area is defined for each PM. Memory address range checking is used at runtime to trap accesses outside these two areas when a memory location is written to or read from in a PM.

label **dsRAM** number of bytes - This allocates storage of specified number of bytes in RAM.

label **dsEEP** number of bytes - This allocates storage of specified number of bytes in EEPROM.

(e) Data Transfer Instructions

Data transfer uses immediate or extended addressing.

LDA[A or **B]** immd or address - Loads value into A or B register. Parameter immd is an immediate value which is a number preceded by an "#". Alternatively a sixteen bit address is specified. This loads an sixteen or eight bit value into the referred register.

STA[**A** or **B**] address - Stores an eight or sixteen bit value from a register to the specified address.

LDD immd or address - Loads value into D register. Parameter immd is an immediate value which is a number preceded by an "#". Alternatively a sixteen bit address is specified. This loads an sixteen or eight bit value into the referred register.

STD address - Stores an eight or sixteen bit value from D register to the specified address.

LDX immd or address - Loads value into IX register. Parameter immd is an immediate value which is a number preceded by an "#". Alternatively a sixteen bit address is specified. This loads an sixteen or eight bit value into the referred register.

STX address - Stores an eight or sixteen bit value from IX register to the specified address.

LDY immd or address - Loads value into IY register. Parameter immd is an immediate value which is a number preceded by an "#". Alternatively a sixteen bit address is specified. This loads an sixteen or eight bit value into the referred register.

STY address - Stores an eight or sixteen bit value from IY register to the specified address.

In order to run a PM, a PM file must have the format as shown in Table 4.

C4D7h
Option
Code Size
Pm Code
CRC

Table 4 PM file layout.

The header identifier is arbitrarily chosen to be C4D7 hexadecimal. The option byte specifies options such as whether a IPIN is required and if the code has access rights to the User Data Area in the EEPROM 316. The CRC will be CRC-CCITT and it will be calculated on the file area starting at the end of the header to the end of PM code.

A PM has to be loaded from its storage area into an execution buffer in the RAM 32 before it can be run. For internal PMs, this is performed by the Operating System. External PMs on Smartcards require:-

- An interface PM residing at a fixed location on the Smartcard. This can also be the only PM on the card. The PM must reside in an area that has no read protection attribute set.

- This interface PM can call another PM. Usually, this is done through a menu setup by the interface PM. It is important that when the last PM in the chain terminates, it calls the ExitPM command to return to the System Main Menu.

- To ensure that the interface PM is valid, a header preceding it is used for verification. Additionally, the CRC attached to the last two bytes of a PM "file" is used to verify its validity before it is run by the Operating System.

In the present embodiment, PMs will only be supported on the MCOS16K, ME2000 and the EEK series Smartcards. The location of the boot up PM is not currently defined.

As previously described, a routine invoking an auto-detection procedure is required to be run to determine the Smartcard type before an attempt can be made to locate the boot up PM.

If a PM calls another PM which requires a secret code (or similar) to be presented, then it is advisable that it be entered form the keypad. Encoding it into a PM will compromise security. In this respect, there are principally two types of Smartcards, namely synchronous cards and asynchronous cards.

Most synchronous Smartcards are basically a serial memory device which do not have commands as such. Normally they do not have read/write control attributes and only some have access control. Hardware driver routines are needed to do the bit by bit accessing of these types of Smartcards.

All asynchronous Smartcards have a microprocessor therein which runs a program that supports storing and retrieving data, usually with some kind of read/write control attributes that are optional and reasonably flexible. Most of these cards have security features that control access to the card, and/or specific data areas, and that enable the changing of passwords. A universal asynchronous receiver transmitter (UART) is needed to communicate with the card. Asynchronous cards issue what is known as the "answer to reset" message in response to resetting of the card, which contains basic information about the link level communications encoding and parameters, programming voltage and current, password retry limits, "MASK" number etc. Asynchronous Smartcards have commands to Create, Open, Close, Read and Write to storage areas, as well as to present a password or PIN code. Some even have DES (Decryption Encryption Standard) cryptographic facilities. Furthermore, one model of card has the ability to store a number of "user commands" written in the native code of the microprocessor.

The relevant subroutine for establishing communications between the hand held device 11 and the IC Smartcard will need to operate the microcomputer 29 in a manner so that it reads some "data" stored on the IC card received within the Smartcard receptor 19, which describes how to treat the Smartcard. This step will have to be used in conjunction with methods to identify a card's type and make. It is necessary to identify the make because although international standard ISO 7816-3 specifies the message format, command codes that perform similar functions are not the same between different makes of asynchronous Smartcards. Other differences that arise between different makes are the format and presentation of passwords and protocols for transferring data. Indeed, at present it is believed that there are three different protocols adopted for transferring data between different makes of Smartcard.

As shown in figure 4, the routine 81 for establishing communication, automatically, involves finding out whether the Smartcard is asynchronous at block 83, synchronous at block 85 or mute at block 87.

In order to determine whether the card is asynchronous 83, the routine uses the services of the Smartcard Interface driver 41 to enable the Clock contact of the Smartcard and to allow the Smartcard out of Reset. If the card is asynchronous, the "answer to reset" message will be sent out automatically by the card. If no answer is received after about one second, then it is assumed that the card is not asynchronous and the program proceeds to testing for whether the card is synchronous or not at block 85.

In order to test for a synchronous card, the routine uses the Smartcard Interface driver 41 to attempt to read some data from the "configuration file" of the IC card. If recognisable data is read, then it is assumed to be a synchronous card and the card is then worked with by reading the entire configuration file as shown at block 89.

If the synchronous test fails, then it is assumed that the card is mute and the routine causes a mute card message to be displayed on the LCD display 17 as represented at block 57.

If the card is found to be asynchronous, the routine needs to proceed with further testing in order to determine whether the make of card falls within the range of card makes supported by the device. The reason for this is that the "answer to reset" message from an asynchronous IC card cannot be used to reliably determine the card make, since the "answer to reset" message does not include any information about the commands and their parameters for the Smartcard which is needed to differentiate between different makes of Smartcard.

For this information, the interface PM of the Smartcard needs to be accessed which has information about the card commands, access control attributes, storage format, data representation methods, and/or other PMs that will be executed by the device 11. In this regard, access control means passwords, PIN codes and methods of transferring, changing and reactivating them; storage format means file names, file sizes and structures, record sizes and structures and number of records; and data representation means what each bit or byte or number of bytes represents for a specific application. Moreover, because storage space on IC cards is limited to an average of 2 kilobytes or less, data will usually be stored in a packed, compressed or token like manner and so data is stored differently to how it is actually displayed.

Catering for the application specifics of Smartcard uses presents special problems as a Smartcard does not run an application program as such, but principally is a means of storing data. Consequently, the Smartcard itself acts under the control of an external device, in the present case being the hand held device 11, and is a physical half duplex device that takes commands and responds to them.

To handle this the PM(s) of the card will be interpreted by the device in a manner that allows data entry, storage and display to be performed, along with password entry, user and Smartcard authentication and authorisation of transactions. If a Smartcard is used for more than one application and by more than one service provider, then there may be a different storage format and/or data representation for the storage area for each application. In this case, the Smartcard may have two levels of PM, the first giving information about the card commands and access control attributes of each data storage area, and the second, which will be inside a storage area, will give information on how to handle the data inside that storage area.

Thus it can be seen that it is important that the device be able to differentiate not only between different card types, but also between different makes and have a way of allowing the device to handle different Smartcard applications. By adopting this method, the device is able to actively work with a selected range of Smartcards from different manufacturers that may all contain data for different and or multiple applications. This method provides a way whereby the device requires no knowledge of the applications specific aspects of the structure, format, attributes and representation methods of the data stored on the Smartcard, so long as the particular make of the Smartcard is supported by the device.

After receiving the answer to reset message, to determine that the IC card is an asynchronous card the routine uses the Smartcard Interface driver 41 to continue to test whether the card is of a make supported by the device, at block 91. This test involves the driver issuing an "open interface PM file" command to the Smartcard, which command corresponds to a specific make of card. If the status is good, then the process will go to the next stage which will involve reading the interface PM file of the card as represented at block 93. If the status is bad, an "open file" command for the next make of card supported by the device is tried, until a good status is returned or there are no more makes or brands of card supported by the device to try. If there are no more brands, then the routine causes an unusable card message to be displayed on the LCD display 17, as represented at block 95.

In order to read the interface PM file as represented at block 93, a "read file" command is issued by the driver for the make of card determined. The data in that file is then used to determine the parameters and the other PMs that need to be loaded and interpreted by the device, so as to access and work with the other data on the Smartcard, as represented at block 97.

Now having described the various features of the hand held device 11, the general method of operating the device will now be described.

The user operating the card will firstly open the card so that the wings 12a and 12b of the housing are divergingly disposed in a planar arrangement as shown in figures 1a to 1c of the drawings. The on/off key 28 is pressed to power up the device if it is off and the operating system 53 executes the main menu 51 and invokes the menu system routine to enable the user to operate the menu. Viewing and selection of the menu items are as previously described using the arrow keys 49a and 49b, the cancel key 50 and the enter key 48.

As previously described, the main menu items which may be selected and displayed are: tone dialler, card terminal, internal PM, card PM and change PIN.

As previously discussed, any of the menu choices can be PIN protected. This option is set via configuration of the device at the time that the device is initialised.

If an item is PIN protected, the prompt "PIN:" appears when the item is seleted. The user is then required to enter the current IPIN, each digit being echoed as an *, and then press the enter key 48. In the present invention the PIN is limited to four digits, whereby any entry after the fourth is ignored. If the IPIN is incorrect, the message "PIN CORRECT" is displayed for three seconds and the user is returned to the main menu.

At the fourth successive incorrect IPIN entry, the device becomes locked and will be unuseable until an initilisation load is performed. This requires a correct password to be presented to he device via the serial communication port in order to commence an initialisation load before the IPIN retry counter can be reset, unlocking the device, as previously described. Upon the device becoming locked, the following message is displayed "PIN LOCKED".

If the PIN is correct, the relevant action is performed as selected by the menu item previously displayed.

In the case of the **tone dialler**, upon selecting this option, the user can key in a sequence of up to sixteen digits, which are echoed on the screen. Pressing the enter key 48 causes the device to generate the corresponding DTMF tone dialling segments. The user can enter another set of digits and repeat the operation as required. Pressing the cancel key 50 causes a return to the main menu.

In the case of the selection of the **Smartcard terminal** item, when the user presses the enter key 48 at this option, the display changes to "READY....". At this stage, the device should be connected to the host device and the user should insert an ISO Smartcard. The hose device can then read from and write to the Smarcard. The device operates in pass through mode, meaning that commands and responses are transmitted unchanged between the Smartcard and the host device.

In the case of the **internal PM** item being selected, when the user presses the enter key 48, a sub-menu of three items is activated, the first displaying "PM 1". The other two items are PM 2 and PM 3. A PM is run by pressing the enter key at the relevant display.

Pressing the cancel key in the sub-menu causes a return to the PM menu, and pressing the cancel key at that menu causes a return to the main menu. If there are no PMs loaded, or if there is none for a particular sub-menu, the message "NO PM" is displayed. After three seconds, the main menu reappears.

Selecting the **Smartcard PM** item, a Smartcard containing a PM should be inserted into the Smartcard reader/writer receptor 19. On pressing the enter key 48, selecting the item, the PM stored upon the card will be loaded into the execution buffer in the device RAM 32 and executed. The first (or only) PM on a Smartcard is called the interface PM and will always be run first. The interface PM could then present the user with a menu that allows other PMs on the card to be run.

If there is no Smartcard inserted, of if there is no PM on the Smartcard, the message "NO PM" appears for three seconds, followed by a return to the main menu.

In the case of selecting the **change PIN** item, the prompt "OLD PIN:" appears. Accordingly, the old PIN must be input, as previously described, and if correct, the user is prompted for the new PIN by the following message appearing "NEW PIN:". The user is then required to enter the new IPIN (which is echoed on the screen for confirmation) and presses the enter key. Consequently, the new IPIN replaces the old and the main menu returns.

It should be noted that an important feature of the hand held device of the present embodiment is that the device is to be programmed via program modules (PMs). These PMs will actually be written externally of the device, and will be supplied for subsequent storage internally within the device or on a Smartcard. There are three stages in the production of a PM:

1. Write source code in a specially devised high level security language. In the present embodiment the language chosen is INTELLECT (trade mark) high level security language (IHSL), which is a BASIC like language.

2. Compile the IHSL source code into PML assembler source code.

3. Pass the assembler source code through a PML assembler to produce the actual PM object code (that is, the executable code).

The use of a three stage process (rather than compiling directly to PM object code) allows optomisation of the PML assembler code and aids trouble shooting during development. Developing PMs can be performed through a development platform, which can provide a menu based, mouse driven integrated environment for the development of PML applications, and includes a run time simulator.

The hand held device 11 of the present embodiment has many applications. Possible functions that the device can perform using customised PMs include:

- Identifying the user via telephone or modem by way of a PM stored internally or on a Smartcard. A user enters the IPIN to activate the ID function then enters a challenge value given over the telephone by a service provider. The device performs cryptographic processing using a secret key also known to the service provider's computer and reads out the result. If the value matches that calculated by the service provider, the user and device are positively identified.

  This function could also be carried out via a modem, in which case the challenge and response would be transmitted over the serial port, or via DTMF.

- Selecting and dialling telephone numbers stored internally or on a Smartcard. The user selects a name by scrolling through a list, then places the device speaker against the telephone handset microphone to dial the number. Entries may be edited by the user.

- Viewing by the user of certain data stored internally or on a Smartcard.

- Making payments from a financial Smartcard for goods or services via telephone or modem. With a financial Smartcard inserted, the user could have a payment deducted over the telephone (via DTMF) or modem.

- Adding value to a financial Smartcard from a bank account via telephone or modem. Funds could be transferred to a financial Smartcard from a remote institution.

- Performing data transfer between a Smartcard and host device such as a personal computer (PC), modem or terminal at a point of service where there is not a Smartcard terminal. This would be useful where the volume of transactions did not warrant the expense of a conventional terminal.

- Functioning as an electronic purse or wallet, whereby a Smartcard could contain details of various on-line accounts (such as are available via EFTPOS) and the user could operate on an account via a telephone (modem or DTMF) connection.

- On course betting, whereby an organisation such as the TAB (trade mark) could issue a betting Smartcard to punters. The user makes bets via menu choices on the OSA/MicroBank and these are written to the Smartcard. After completing all bets, the user puts the card in an on-course Smartcard receptacle so that the bet details can be read by the TAB computer and an authentication value written back to the card. A card with winning bets could be presented to a teller, who would place the card in a receptacle and receive a display of the amount to be paid. The card would be retained for re-use by the TAB.

The hand held device can perform many other kinds of transactions outlined above, depending upon the level of security required, either by telephone or modem, by terminal or as stand alone.

When connecting to a modem or using tone signalling via telephone, the device would be communicating to a host computer system and transactions would be performed on-line. When connecting to a terminal, transactions may be performed on-line or off-line from the host computer system, depending on the requirements.

In the case of stand alone, transactions would be totally off-line and may involve removing the user's Smartcard and inserting the provider's Smartcard.

At present, there is only a draft ISO specification that attempts to formalise the basis for using IC cards for financial applications based on the implementation of ISO 7816 part four. The latter is attempting to specify standard command types and in the usage of the elements of the message structure for IC cards. Unfortunately, the finalisation of this standard is not moving particularly fast, principally due to different IC card makers attempting to have their particular command set adopted as the standard. Although the ISO specifications do not address the need to provide information about the format of data, how it is stored or used and what the data means, there is nonetheless a logical data element of the structure defined in DIS 9992. This logical data element if and when it is standardised can also be utilised by the hand held device to enable it to attach information to a card or file to indicate the application specific parameters for the card and/or device. Accordingly, this is seen as a longer term use of the device for the future.

It should be appreciated that the scope of the present invention is not limited to the specific embodiment herein described, and accordingly other embodiments incorporating only some or all of the parts of the described embodiment and/or incorporating modifications in accordance with common general knowledge are considered to fall within the scope of the present invention.

For example, in an alternative embodiment, the housing is of monolithic form as opposed to comprising two hinged wings, whereby the IC card receptor is integral with one side of the housing and the keypad and liquid crystal display is integrated with the other side of the housing.

In a further embodiment, the housing may include a slide out IC card receptor for receiving the IC card at the rear of the device and incorporating either the microphone or loud speaker, the other of which is disposed at the other end of the card so that when the slide out IC card receptor is fully extended, the distance between the microphone and speaker is commensurate to the distance between the microphone and speaker of a telephone handset to facilitate DTMF communications, in the manner previously described.

Claims:-

1. A device for communicating with an integrated circuit card comprising-

a user interface;

an external interface including an IC card interface for connectedly receiving and communicating with an IC card;

microcomputer means having internal storage for storing data said microcomputer means being connected to said user interface and said external interface; and

an operating system for operating said microcomputer means to control said internal storage, said user interface and said external interface;

wherein said data includes a program module loaded into said internal storage for execution upon activation via said user interface, said program module being loaded from a said IC card or from a host connected to the device.

2. A device as claimed in claim 1, including an interpreter for executing a said program module, wherein said interpreter operates as a virtual machine on top of said microcomputer means and is adapted to prevent access to certain secured areas of said internal storage and said microcomputer means, and to prevent the direct execution of native program code of said microcomputer means.

3. A device as claimed in claim 2, wherein said interpreter comprises a compiler means for converting high level program module language into program module object code in accordance with a prescribed instruction set.

4. A device as claimed in claim 3, wherein said compiler means comprises a compiler for converting said high level program module language into program module language assembler source code, and an assembler for converting said program module language assembler source code into said program module object code, in accordance with respective prescribed instruction sets.

5. A device as claimed in any one of the preceding claims, wherein said external interface includes a serial communications interface for allowing full duplex serial communications between said microcomputer means and said host connected thereto.

6. A device as claimed in any one of the preceding claims, wherein said operating system provides for one or more communication options when a said IC card is connected to, said IC card interface, said IC card interface providing services with said serial interface or said user interface in conjunction with said microcomputer means to achieve said communication option.

7. A device as claimed in claim 6, wherein one said communication option comprises loading a said program module directly from said IC card for execution.

8. A device as claimed in claim 7, wherein said program module is loaded into a temporary storage area and executed upon said activation.

9. A device as claimed in any one of claims 6 to 8, wherein another said communication option comprises passing messages between said serial interface when a host is connected thereto, and said IC card interface when an IC card connected thereto, under the control of said microcomputer

means and said operating system upon activation via said user interface.

10. A device as claimed in any one of the preceding claims, wherein said external interface includes a microphone for receiving tone signals remotely transmitted to the device, a speaker for generating tone signals to be transmitted from the device, and a tone signalling interface interconnecting said microphone, said speaker and said microcomputing means for decoding tone signals received by said microphone and for driving said speaker, whereby said operating system provides for a further communication option using said tone signalling interface with a telephone system upon activation via said user interface.

11. A device as claimed in any one of the preceding claims, as dependent upon claim 5, wherein said internal storage comprises a semi-permanent storage area for storing data loaded from a host connected to said serial communications interface.

12. A device as claimed in claim 11, including a system loader to receive and execute prescribed commands via said serial communications interface for loading said data, wherein said system loader prevents the direct execution of native program code of said microcomputer means.

13. A device as claimed in claim 11 or 12, wherein a said program module loaded from a said host connected to said serial communications interface is stored in said semi-permanent storage area, and wherein said operating system is adapted to load a said program module stored in said semi-permanent area into said temporary area for execution upon said activation.

14. A device as claimed in any one of the preceding claims as dependent upon claim 6, wherein said IC card interface includes a routine for determining the compatibility of said IC card with the device and configuring said IC card interface for communicating with a compatible IC card connected thereto.

15. A device as claimed in claim 14, wherein said routine defines different subroutines to enable the device to differentiate between different types and makes of IC cards.

16. A device as claimed in any one of the preceding claims, wherein said operating system provides for an option of a personal identification number subroutine to be performed to enable said microcomputing means to validate the identity of a user of the device before allowing a predetermined communication option to be activated by a user.

17. A device as claimed in any one of the preceding claims, wherein said internal storage comprises a core area for permanently storing firmware for said operating system, said user interface and said external interface, said firmware being generally divided into three layers: the first layer comprising system and driver firmware, including said operating system and said external interface for direct hardware interfaces and firmware task management; the second layer comprising application firmware, including said user interface, for supporting the functional requirements of the device; and the third layer comprising program module interpreter firmware for supporting the operation of a said program module.

18. A device as claimed in any one of the preceding claims as dependent upon claim 11, wherein said semi-permanent

storage area also provides for storage of: system information data including a password for an initialisation loading access, an internal personal identification number (IPIN) for user access; cipher keys for cryptographic processing; program module control data; and function parameters including configuration data for the device.

19. A device as claimed in any one of the preceding claims as dependent upon claim 8, wherein said temporary storage area is partitioned into areas for running the firmware, program module volatile data areas and intermediate storage for cryptographic functions performed by the device.

20. A portable hand held device for communicating with an integrated circuit card comprising:-

a housing;

a keypad disposed on one portion of said housing;

a display disposed on another portion of said housing; and

microcomputing means disposed within said housing and connected to said keypad and said display for receiving input signals provided by said keypad, and sending output signals for display by said display, said microcomputing means having a computer program for operating said microcomputing means in accordance with a prescribed routine to communicate with a user of the device;

wherein a self-contained power source is disposed within said housing for supplying power to said processing means;

an integrated circuit card receptor is disposed on a further portion of said housing and connected to said processing means for receiving an integrated circuit card and communicating therewith in accordance with said prescribed routine and the input signals provided by said keypad; and

said housing is of a size sufficiently small to be carried on one's person.

21. A device substantially as herein described with reference to the accompanying drawings.

22. A method of operating a device as claimed in any one of claims 1 to 21.

23. A method for communicating with an integrated circuit card comprising the steps of:-

testing the integrated circuit card to determine whether it is one of a predetermined range of integrated circuit card types;

issuing a prescribed command or commands to the integrated circuit card corresponding to a particular make of integrated circuit card if said testing determines that said integrated circuit card is one of a prescribed type for communicating;

checking the response of the integrated circuit card to said issuing to determine whether the response is in accord with the expected response prescribed for said particular make;

repeating steps 2 and 3 using other prescribed commands to the integrated circuit card corresponding to other particular makes of integrated circuit if said response is not in accord with the expected response prescribed for said particular make corresponding to the prescribed command or commands until a prescribed make or integrated circuit card is determined; and

reading the configuration file using the prescribed commands corresponding to the particular make of integrated circuit card if said response is in accord with the expected response prescribed for said particular make to establish the parameters and language for accessing and working with the integrated circuit card.

24. A method substantially as herein described.

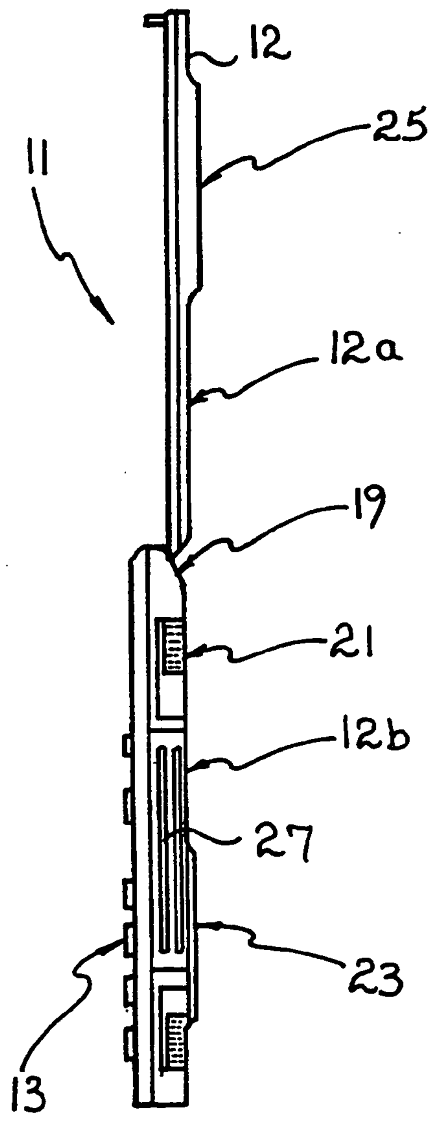


Fig. 1A.

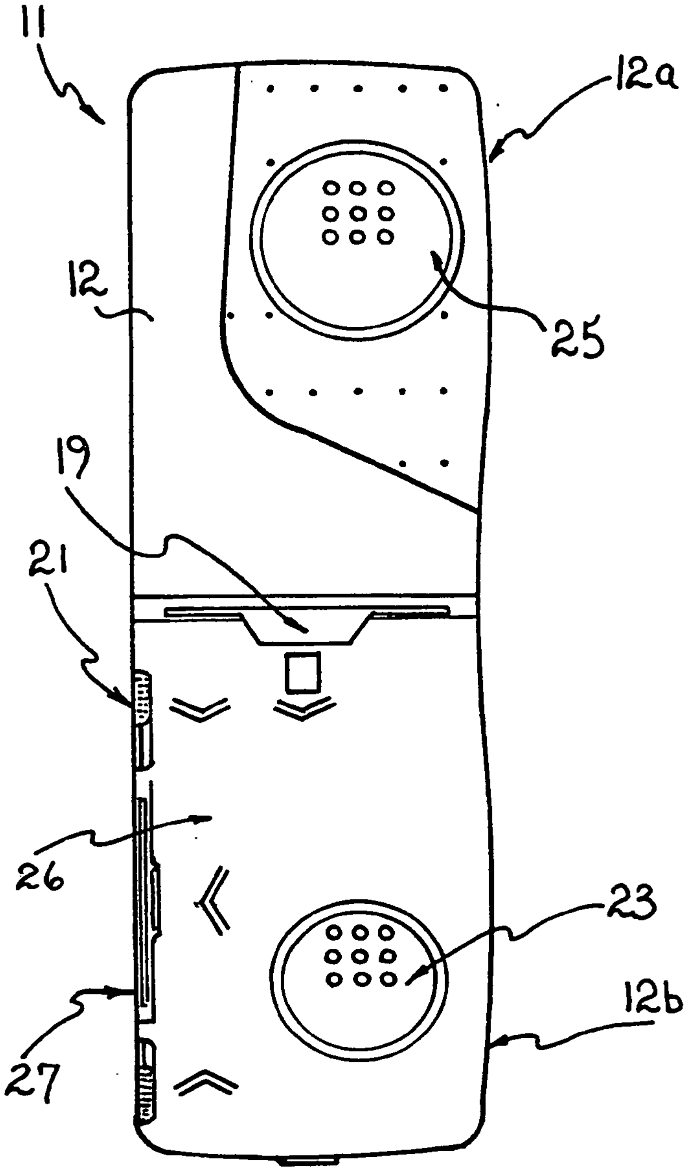


Fig. 1B.

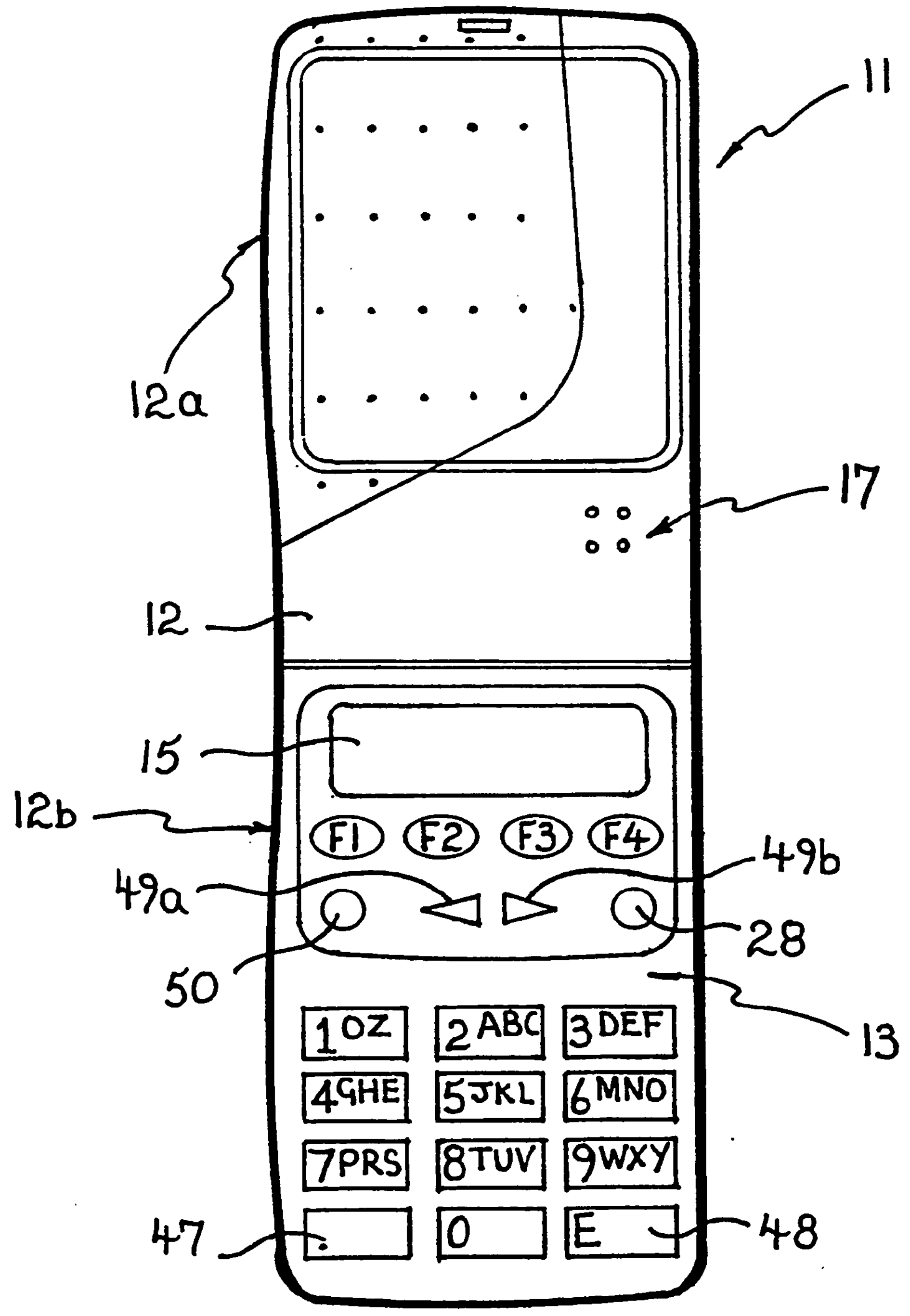

11
12a
17
12
15
12b
F1
F2
F3
F4
49b
49a
28
50
13
1 OZ
2 ABC
3 DEF
4 GHE
5 JKL
6 MNO
7 PRS
8 TUV
9 WXY
0
E
47
48

Fig. 1C.

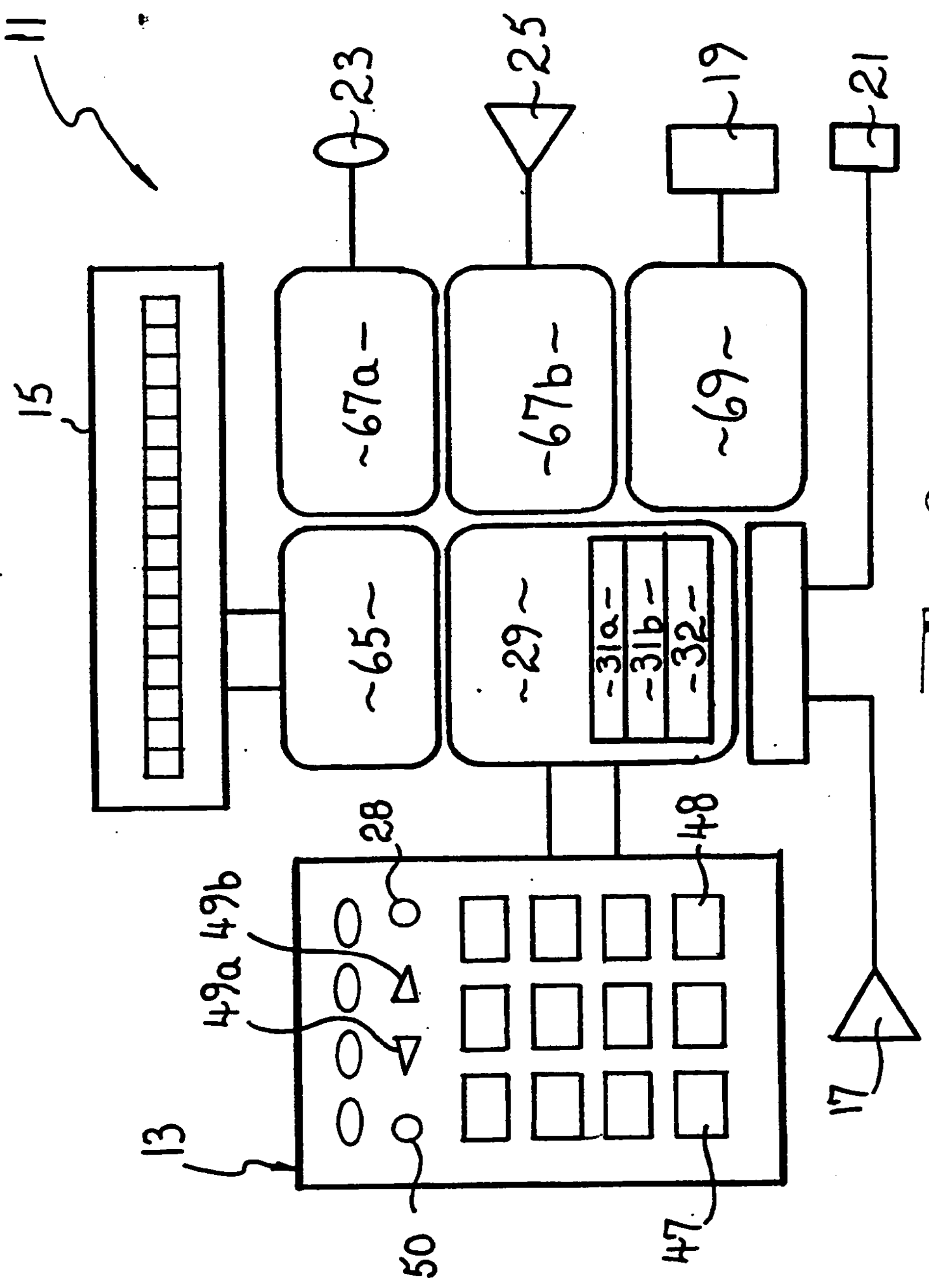

11
15
13
17
19
21
23
25
28
~29~
~31a~
~31b~
~32~
47
48
49a
49b
50
~65~
~67a~
~67b~
~69~

Fig 2

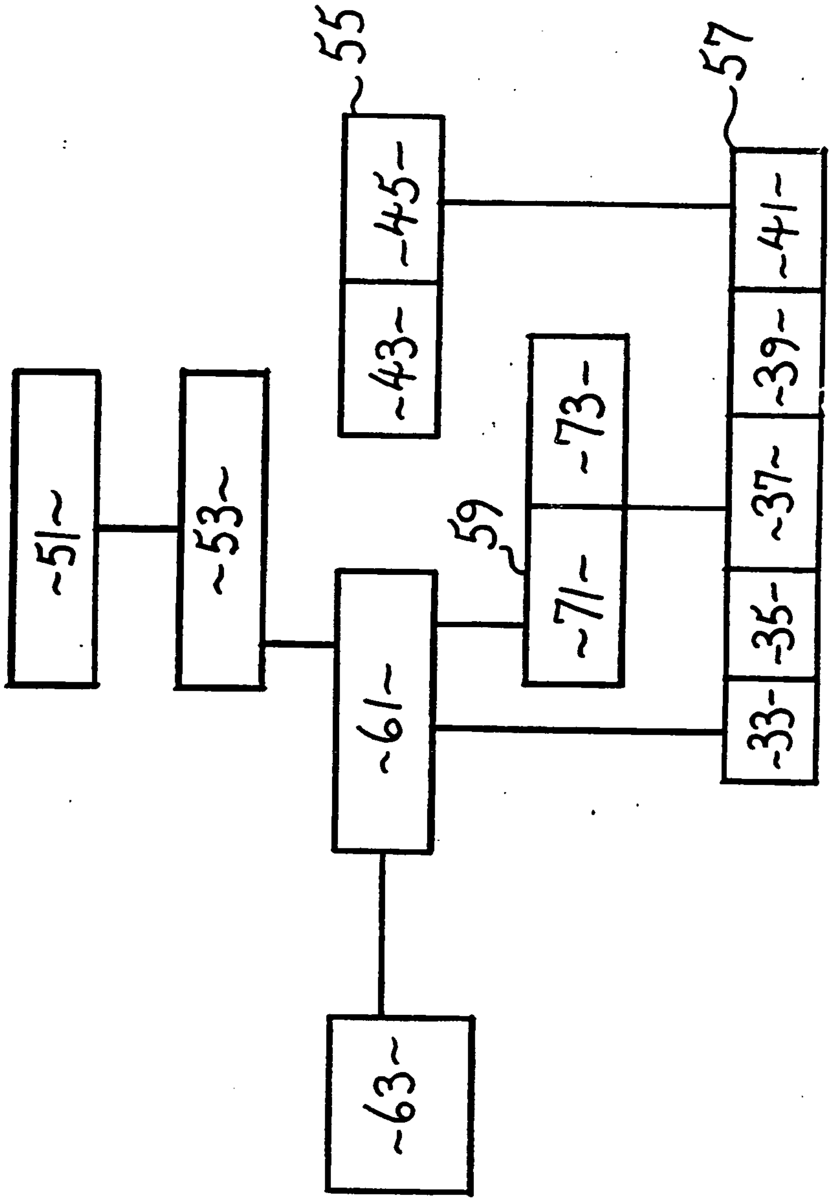


Fig. 3

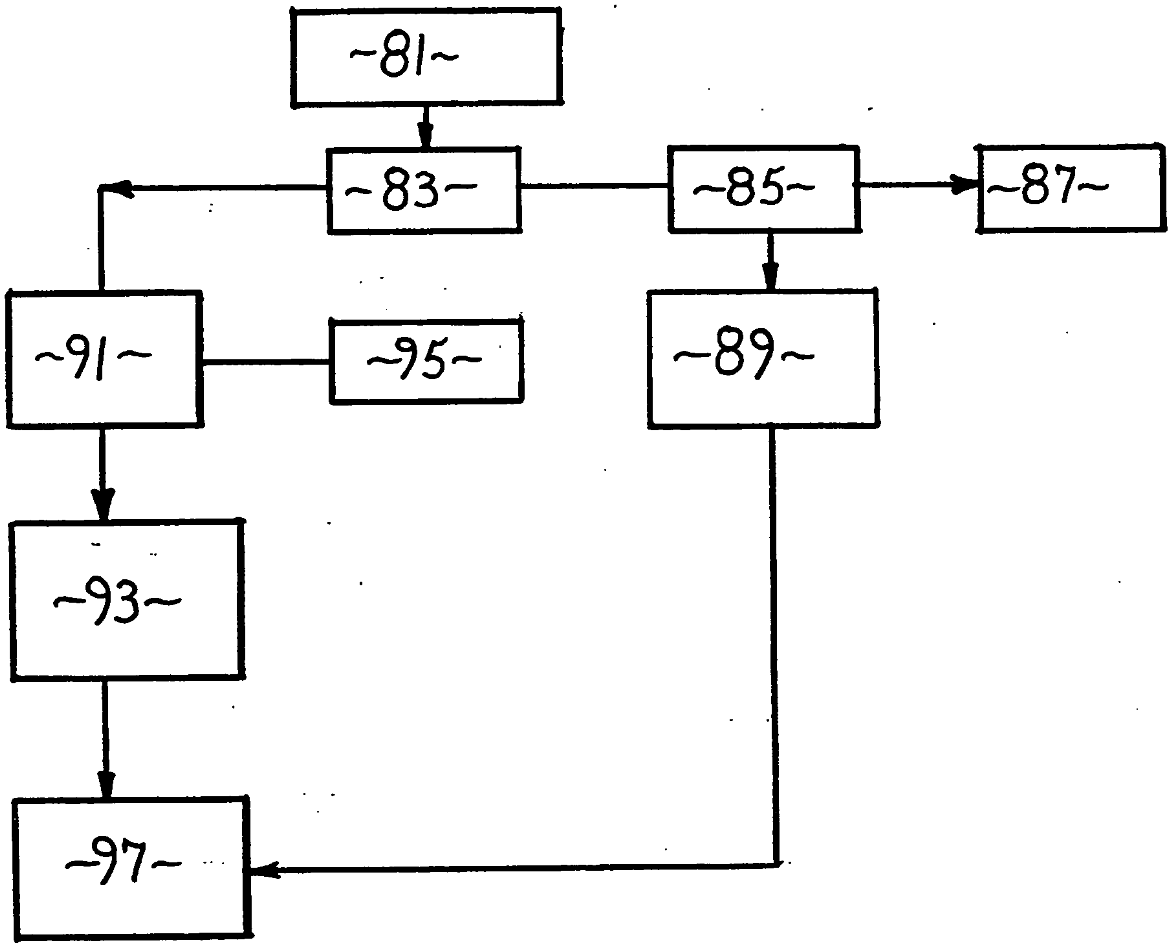


Fig. 4.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU 94/00437

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl.6 G06K 19/07

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC : G06K 19/07

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
AU : IPC as above

Electronic data base consulted during the international search (name of data base, and where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to Claim No.
X	EP,A, 490455 (THOMSON CONSUMER ELECTRONICS SA) 17 June 1992 (17.06.92)	1-24
X	GB,A, 2250361 (MITSUBISHI DENKI KK) 3 June 1992 (03.06.92)	1-24
X	FR,A, 2667171 (GEMPLUS CARD INTERNATIONAL) 27 March 1992 (27.03.92)	1-24
A	EP,A, 486363 (THOMSON CSF) 20 May 1992 (20.05.92)	

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

*	Special categories of cited documents :
"A"	document defining the general state of the art which is not considered to be of particular relevance
"E"	earlier document but published on or after the international filing date
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O"	document referring to an oral disclosure, use, exhibition or other means
"P"	document published prior to the international filing date but later than the priority date claimed
"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&"	document member of the same patent family

Date of the actual completion of the international search 27 October 1994 (27.10.94)	Date of mailing of the international search report 8 Nov 1994 (8.11.94)
Name and mailing address of the ISA/AU AUSTRALIAN INDUSTRIAL PROPERTY ORGANISATION PO BOX 200 WODEN ACT 2606 AUSTRALIA Facsimile No. 06 2853929	Authorized officer John Thomson J W THOMSON Telephone No. (06) 2832214

INTERNATIONAL SEARCH REPORT
Information on patent family membe.

International application No.
PCT/AU 94/00437

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member			
EP	490455	JP	5210765		
GB	2250361	DE	4137336	JP	4178791
EP	486363	FR	2669452		
					END OF ANNEX